OpenStreetMaps Project- Mapping Trenton, Princeton & Plainsboro City Area, New Jersey

Udacity Data Science Nanodegree course- Final Project

Data Wrangling with Mongo DB

By: Mubina Arastu

MArastu@wgu.edu

Summary

This project details my analysis and cleanup of the OpenStreetMap.org map data for the majority of Trenton, Princeton & Plainsboro cities of New Jersey. I focused on street addresses and amenities listed within the map.I choose these cities because I live and work in these areas.

This writeup list out few problems that were encountered in the dataset, and gives an high level overview of the data. Exploratory data analysis, wrangling and cleaning of this project can be seen in python notebook.

Problems With the Dataset

Initialy I faced difficulty in downloading a reasonable size dataset. After few attempts, I finally downloaded OSM file that included three cities Trenton, Princeton and Plainsboro.

As per my exploratory analysis of the dataset, I found few issues that needed to be cleaned up:

1. Street types and names to clean up.
2. Brand names doesn't have a 'type' tag. Wikipedia tag need to be updated as type.
3. Amenities like shops have some bad data that need to be cleaned up.
4. Companies & Healthcare have some problamatic data that need to be cleaned up

Street Address Cleanup and Brand Names cleanup can be seen in 'Trenton_Princeton_Maps'.ipynb file

Check Tag Types & Map Area Boundaries

In [102]:
```
import xml.etree.ElementTree as ET
import pprint

def count_tags(filename):
    tags = {}
    bounds = {}
    for event, elem in ET.iterparse(filename):
        if elem.tag == "bounds":
            for latlong in elem.attrib:
                bounds[latlong] = elem.attrib[latlong]
        if elem.tag in tags.keys():
            tags[elem.tag] += 1
        else:
            tags[elem.tag] = 1
    return tags, bounds

with open('trenton-princeton.xml', 'rb') as mapfile:
    tags, bounds = count_tags(mapfile)
    print ("Found number oftags:")
    pprint.pprint(tags)
    print ("\nMap Area Boundaries:")
    pprint.pprint(bounds)
```

```
Found number oftags:
{'bounds': 1,
 'member': 162351,
 'meta': 1,
 'nd': 831423,
 'node': 686750,
 'note': 1,
 'osm': 1,
 'relation': 1653,
 'tag': 328515,
 'way': 80213}

Map Area Boundaries:
{'maxlat': '40.3887000',
 'maxlon': '-74.4464000',
 'minlat': '40.1860000',
 'minlon': '-74.8543000'}
```

Check k-values

Below code is to check if there are any k values with special characters and also to see K:Ktype colon seperated pairs. Below code is to check if there are any k values with special characters and also to see K:Ktype colon seperated pairs.

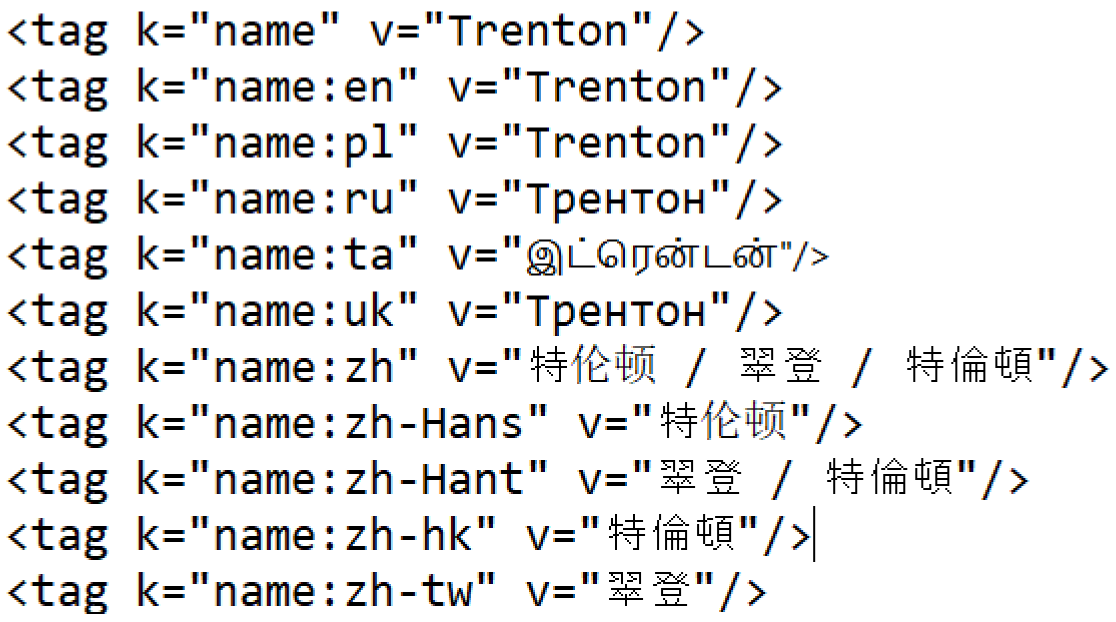
In [104]:
```
import xml.etree.ElementTree as ET
import pprint
import re

lowercase_characters = re.compile(r'^([a-z]|_)*$')
lowercase_colon = re.compile(r'^([a-z]|_)*:([a-z]|_)*$')
specialchars = re.compile(r'[=\+/&<>;\'"\?%#$@\,\. \t\r\n]')
lowercase_colon_vals = {}

def key_type(element, keys):
    if element.tag == "tag":
        kval = element.attrib['k']
        if re.search(lowercase_characters, kval):
            keys['lower case'] += 1
        elif re.search(lowercase_colon, kval):
            keys['lower case & colon values'] += 1
            colvals = kval.split(':')
            if colvals[0] not in lowercase_colon_vals.keys():
                lowercase_colon_vals[colvals[0]] = set()
            lowercase_colon_vals[colvals[0]].add(colvals[1])
        elif re.search(specialchars , kval):
            keys['special characters'] += 1
        else:
            keys['other'] += 1

    return keys


def process_map(filename):
    keys = {"lower case": 0, "lower case & colon values": 0, "special characte
rs": 0, "other": 0}
    for _, element in ET.iterparse(filename):
        keys = key_type(element, keys)

    return keys

with open('trenton-princeton.xml', 'rb') as mapfile:
    keys = process_map(mapfile)
    print ("Types of k-values and their counts:")
    pprint.pprint(keys)
    print ("Types of colon-separated k-values:")
    pprint.pprint(lowercase_colon_vals)
```

```
Types of k-values and their counts:
{'lower case': 204450,
 'lower case & colon values': 114110,
 'other': 9955,
 'special characters': 0}
Types of colon-separated k-values:
{'abandoned': {'highway', 'railway'},
 'access': {'forward', 'delivery'},
 'addr': {'city',
          'country',
          'county',
          'full',
          'housename',
          'housenumber',
          'place',
          'postcode',
          'state',
          'street',
          'suite',
          'unit'},
 'aerodrome': {'type'},
 'aircraft': {'type'},
 'alt_name': {'be', 'vi'},
 'alt_short_name': {'pl', 'en'},
 'area': {'highway', 'aeroway'},
 'artist': {'wikidata'},
 'authentication': {'none', 'app', 'nfc'},
 'beacon': {'frequency', 'type', 'code', 'channel'},
 'brand': {'wikipedia', 'wikidata'},
 'bridge': {'ref', 'support', 'name'},
 'building': {'year_built', 'levels', 'flats', 'use', 'colour', 'part'},
 'capacity': {'disabled'},
 'census': {'population'},
 'charging_station': {'output'},
 'check_date': {'surface'},
 'colour': {'arrow', 'text', 'back'},
 'communication': {'aeronautical_mobile_service',
                   'mast',
                   'mobile_phone',
                   'radio',
                   'satellite',
                   'television'},
 'contact': {'email',
             'facebook',
             'fax',
             'foursquare',
             'instagram',
             'name',
             'phone',
             'twitter',
             'website'},
 'crossing': {'barrier', 'island'},
 'cycleway': {'left', 'both', 'right'},
 'dance': {'teaching'},
 'description': {'en'},
 'destination': {'lanes', 'street', 'ref', 'symbol'},
 'diet': {'vegetarian', 'vegan'},
 'disused': {'amenity', 'railway', 'shop', 'building'},
 'drink': {'juice'},
 'fire_hydrant': {'type'},
 'flag': {'type'},
 'fuel': {'cng', 'biodiesel', 'diesel', 'lpg', 'biogas'},
 'garden': {'type'},
 'generator': {'source', 'type', 'method'},
 'gnis': {'county_id',
          'county_name',
          'cr',
          'created',
          'edited',
          'feature_id',
          'feature_type',
          'id',
          'import_uuid',
          'reviewed',
          'state_id'},
 'gtfs': {'agency_id'},
 'healthcare': {'counselling', 'speciality'},
 'heritage': {'operator'},
 'hgv': {'minweight', 'conditional', 'national_network', 'state_network'},
 'internet_access': {'fee', 'ssid'},
 'is_in': {'country',
           'country_code',
           'county',
           'state',
           'state_code',
           'township'},
 'isced': {'level'},
 'junction': {'ref'},
 'kerb': {'height'},
 'lanes': {'both_ways', 'forward', 'backward'},
 'maxspeed': {'advisory', 'freight', 'type'},
 'maxweight': {'signed'},
 'memorial': {'type'},
 'mtb': {'scale'},
 'name': {'ab',
          'ace',
          'af',
          'als',
          'alt',
          'am',
          'an',
          'anc',
          'ang',
          'ar',
          'arc',
          'arz',
          'as',
          'ast',
          'av',
          'ay',
          'az',
          'ba',
          'backward',
          'bar',
              'bcl',
              'be',
              'bg',
              'bi',
              'bm',
              'bn',
              'bo',
              'bpy',
              'br',
              'bs',
              'bxr',
              'ca',
              'cdo',
              'ce',
              'ceb',
              'chr',
              'chy',
              'ckb',
              'co',
              'crh',
              'cs',
              'csb',
              'cu',
              'cv',
              'cy',
              'da',
              'de',
              'din',
              'diq',
              'dsb',
              'dv',
              'dz',
              'ee',
              'el',
              'eml',
              'en',
              'eo',
              'es',
              'et',
              'eu',
              'ext',
              'fa',
              'ff',
              'fi',
              'fo',
              'forward',
              'fr',
              'frp',
              'frr',
              'fur',
              'fy',
              'ga',
              'gag',
              'gan',
              'gd',
              'gl',
              'glk',
                 'gn',
                 'gu',
                 'gv',
                 'ha',
                 'hak',
                 'haw',
                 'he',
                 'hi',
                 'hif',
                 'hr',
                 'hsb',
                 'ht',
                 'hu',
                 'hy',
                 'ia',
                 'id',
                 'ie',
                 'ig',
                 'ik',
                 'ilo',
                 'io',
                 'is',
                 'it',
                 'iu',
                 'ja',
                 'jbo',
                 'jv',
                 'ka',
                 'kaa',
                 'kab',
                 'kbd',
                 'ki',
                 'kk',
                 'kl',
                 'km',
                 'kn',
                 'ko',
                 'koi',
                 'krc',
                 'ks',
                 'ksh',
                 'ku',
                 'kv',
                 'kw',
                 'ky',
                 'la',
                 'lad',
                 'lb',
                 'lbe',
                 'lez',
                 'lg',
                 'li',
                 'lij',
                 'lmo',
                 'ln',
                 'lo',
                 'lt',
                 'ltg',
                 'lv',
                 'lzh',
                 'mdf',
                 'mg',
                 'mhr',
                 'mi',
                 'min',
                 'mk',
                 'ml',
                 'mn',
                 'mo',
                 'mr',
                 'mrj',
                 'ms',
                 'mt',
                 'mwl',
                 'my',
                 'myv',
                 'mzn',
                 'na',
                 'nah',
                 'nan',
                 'nap',
                 'nds',
                 'ne',
                 'new',
                 'nl',
                 'nn',
                 'no',
                 'nov',
                 'nrm',
                 'nso',
                 'nv',
                 'oc',
                 'old',
                 'om',
                 'or',
                 'os',
                 'pa',
                 'pag',
                 'pam',
                 'pap',
                 'pcd',
                 'pdc',
                 'pfl',
                 'pih',
                 'pl',
                 'pms',
                 'pnb',
                 'ps',
                 'pt',
                 'qu',
                 'rm',
                 'rn',
                 'ro',
                 'ru',
                  'rue',
                  'rw',
                  'sa',
                  'sah',
                  'sc',
                  'scn',
                  'sco',
                  'sd',
                  'se',
                  'sg',
                  'sh',
                  'si',
                  'sk',
                  'sl',
                  'sm',
                  'smn',
                  'sms',
                  'sn',
                  'so',
                  'sq',
                  'sr',
                  'srn',
                  'ss',
                  'stq',
                  'su',
                  'sv',
                  'sw',
                  'szl',
                  'ta',
                  'te',
                  'tet',
                  'tg',
                  'th',
                  'tk',
                  'tl',
                  'tn',
                  'to',
                  'tpi',
                  'tr',
                  'ts',
                  'tt',
                  'tw',
                  'ty',
                  'tzl',
                  'udm',
                  'ug',
                  'uk',
                  'ur',
                  'uz',
                  'vec',
                  'vep',
                  'vi',
                  'vls',
                  'vo',
                  'wa',
                  'war',
                  'wo',
                'wuu',
                'xal',
                'xh',
                'xmf',
                'yi',
                'yo',
                'yue',
                'za',
                'zea',
                'zh',
                'zu'},
 'network': {'wikipedia', 'wikidata'},
 'nist': {'state_fips', 'fips_code'},
 'notable_tenant': {'wikidata'},
 'note': {'lanes', 'old_railway_operator'},
 'official_name': {'eo', 'nl', 'en', 'cs', 'din', 'vi', 'pl'},
 'old_name': {'eo', 'ru', 'en', 'vi', 'pl'},
 'old_short_name': {'ru'},
 'operator': {'wikipedia', 'type', 'short', 'wikidata'},
 'osmc': {'symbol'},
 'parking': {'fee'},
 'payment': {'american_express',
             'apple_pay',
             'bitcoin',
             'cash',
             'cheque',
             'coins',
             'contactless',
             'credit_cards',
             'debit_cards',
             'discover_card',
             'ebt',
             'google_pay',
             'litecoin',
             'mastercard',
             'notes',
             'snap',
             'visa',
             'visa_debit',
             'wic'},
 'phone': {'alt'},
 'plant': {'source', 'method'},
 'playground': {'theme'},
 'public_transport': {'version'},
 'radio_transponder': {'category', 'callsign'},
 'railway': {'ref', 'track_ref', 'position', 'historic', 'traffic_mode'},
 'ramp': {'wheelchair'},
 'ref': {'fips',
         'hmdb',
         'njhrp',
         'njrhp',
         'nrhp',
         'penndot',
         'retrofitness',
         'walmart'},
 'restriction': {'hgv', 'conditional'},
 'roof': {'colour', 'shape', 'levels'},
 'short_name': {'nl', 'en', 'ru', 'be', 'zh', 'cs', 'vi', 'es', 'mo', 'pl'},
 'social_facility': {'for'},
 'socket': {'chademo', 'tesla_supercharger'},
 'source': {'alt_name',
            'cycleway',
            'date',
            'geometry',
            'lanes',
            'maxspeed',
            'name',
            'note',
            'old_ref',
            'oneway',
            'population',
            'short_name',
            'surface',
            'unsigned_ref'},
 'swing_gate': {'type'},
 'theatre': {'type'},
 'tiger': {'cfcc',
           'county',
           'name_base',
           'name_direction_prefix',
           'name_direction_suffix',
           'name_type',
           'reviewed',
           'separated',
           'source',
           'tlid',
           'upload_uuid',
           'zip_left',
           'zip_right'},
 'toilets': {'disposal', 'wheelchair'},
 'tower': {'construction', 'type'},
 'traffic_signals': {'direction'},
 'turn': {'lanes'},
 'was': {'amenity'},
 'website': {'state'}}
```

In [113]:
```python
import xml.etree.ElementTree as ET
import pprint
import re

lowercase_characters = re.compile(r'^([a-z]|_)*$')
lowercase_colon = re.compile(r'^([a-z]|_)*:([a-z]|_)*$')
specialchars = re.compile(r'[=\+/&<>;\'"\?%#$@\,\. \t\r\n]')
lowercase_colon_vals = {}

def key_type(element, keys):
    if element.tag == "tag":
        kval = element.attrib['k']
        if re.search(lowercase_characters, kval):
            keys['lower case'] += 1
        elif re.search(lowercase_colon, kval):
            keys['lower case & colon values'] += 1
            colvals = kval.split(':')
            if colvals[0] not in lowercase_colon_vals.keys():
                lowercase_colon_vals[colvals[0]] = set()
            lowercase_colon_vals[colvals[0]].add(colvals[1])
        elif re.search(specialchars , kval):
            keys['special characters'] += 1
        else:
            keys['other'] += 1

    return keys


def process_map(filename):
    keys = {"lower case": 0, "lower case & colon values": 0, "special characte
rs": 0, "other": 0}
    for _, element in ET.iterparse(filename):
        keys = key_type(element, keys)

    return keys

with open('trenton-princeton.xml', 'rb') as mapfile:
    keys = process_map(mapfile)
    print ("Types of k-values and their counts:")
    pprint.pprint(keys)
    print ("Types of colon-separated k-values:")
    pprint.pprint(lowercase_colon_vals)
```

```
Types of k-values and their counts:
{'lower case': 204450,
 'lower case & colon values': 114110,
 'other': 9955,
 'special characters': 0}
Types of colon-separated k-values:
{'abandoned': {'highway', 'railway'},
 'access': {'forward', 'delivery'},
 'addr': {'city',
          'country',
          'county',
          'full',
          'housename',
          'housenumber',
          'place',
          'postcode',
          'state',
          'street',
          'suite',
          'unit'},
 'aerodrome': {'type'},
 'aircraft': {'type'},
 'alt_name': {'be', 'vi'},
 'alt_short_name': {'pl', 'en'},
 'area': {'highway', 'aeroway'},
 'artist': {'wikidata'},
 'authentication': {'none', 'app', 'nfc'},
 'beacon': {'frequency', 'type', 'code', 'channel'},
 'brand': {'wikipedia', 'wikidata'},
 'bridge': {'ref', 'support', 'name'},
 'building': {'year_built', 'levels', 'flats', 'use', 'colour', 'part'},
 'capacity': {'disabled'},
 'census': {'population'},
 'charging_station': {'output'},
 'check_date': {'surface'},
 'colour': {'arrow', 'text', 'back'},
 'communication': {'aeronautical_mobile_service',
                   'mast',
                   'mobile_phone',
                   'radio',
                   'satellite',
                   'television'},
 'contact': {'email',
             'facebook',
             'fax',
             'foursquare',
             'instagram',
             'name',
             'phone',
             'twitter',
             'website'},
 'crossing': {'barrier', 'island'},
 'cycleway': {'left', 'both', 'right'},
 'dance': {'teaching'},
 'description': {'en'},
 'destination': {'lanes', 'street', 'ref', 'symbol'},
 'diet': {'vegetarian', 'vegan'},
 'disused': {'amenity', 'railway', 'shop', 'building'},
 'drink': {'juice'},
 'fire_hydrant': {'type'},
 'flag': {'type'},
 'fuel': {'cng', 'biodiesel', 'diesel', 'lpg', 'biogas'},
 'garden': {'type'},
 'generator': {'source', 'type', 'method'},
 'gnis': {'county_id',
          'county_name',
          'cr',
          'created',
          'edited',
          'feature_id',
          'feature_type',
          'id',
          'import_uuid',
          'reviewed',
          'state_id'},
 'gtfs': {'agency_id'},
 'healthcare': {'counselling', 'speciality'},
 'heritage': {'operator'},
 'hgv': {'minweight', 'conditional', 'national_network', 'state_network'},
 'internet_access': {'fee', 'ssid'},
 'is_in': {'country',
           'country_code',
           'county',
           'state',
           'state_code',
           'township'},
 'isced': {'level'},
 'junction': {'ref'},
 'kerb': {'height'},
 'lanes': {'both_ways', 'forward', 'backward'},
 'maxspeed': {'advisory', 'freight', 'type'},
 'maxweight': {'signed'},
 'memorial': {'type'},
 'mtb': {'scale'},
 'name': {'ab',
          'ace',
          'af',
          'als',
          'alt',
          'am',
          'an',
          'anc',
          'ang',
          'ar',
          'arc',
          'arz',
          'as',
          'ast',
          'av',
          'ay',
          'az',
          'ba',
          'backward',
          'bar',
                 'bcl',
                 'be',
                 'bg',
                 'bi',
                 'bm',
                 'bn',
                 'bo',
                 'bpy',
                 'br',
                 'bs',
                 'bxr',
                 'ca',
                 'cdo',
                 'ce',
                 'ceb',
                 'chr',
                 'chy',
                 'ckb',
                 'co',
                 'crh',
                 'cs',
                 'csb',
                 'cu',
                 'cv',
                 'cy',
                 'da',
                 'de',
                 'din',
                 'diq',
                 'dsb',
                 'dv',
                 'dz',
                 'ee',
                 'el',
                 'eml',
                 'en',
                 'eo',
                 'es',
                 'et',
                 'eu',
                 'ext',
                 'fa',
                 'ff',
                 'fi',
                 'fo',
                 'forward',
                 'fr',
                 'frp',
                 'frr',
                 'fur',
                 'fy',
                 'ga',
                 'gag',
                 'gan',
                 'gd',
                 'gl',
                 'glk',
                    'gn',
                    'gu',
                    'gv',
                    'ha',
                    'hak',
                    'haw',
                    'he',
                    'hi',
                    'hif',
                    'hr',
                    'hsb',
                    'ht',
                    'hu',
                    'hy',
                    'ia',
                    'id',
                    'ie',
                    'ig',
                    'ik',
                    'ilo',
                    'io',
                    'is',
                    'it',
                    'iu',
                    'ja',
                    'jbo',
                    'jv',
                    'ka',
                    'kaa',
                    'kab',
                    'kbd',
                    'ki',
                    'kk',
                    'kl',
                    'km',
                    'kn',
                    'ko',
                    'koi',
                    'krc',
                    'ks',
                    'ksh',
                    'ku',
                    'kv',
                    'kw',
                    'ky',
                    'la',
                    'lad',
                    'lb',
                    'lbe',
                    'lez',
                    'lg',
                    'li',
                    'lij',
                    'lmo',
                    'ln',
                    'lo',
                    'lt',
                                 'ltg',
                                 'lv',
                                 'lzh',
                                 'mdf',
                                 'mg',
                                 'mhr',
                                 'mi',
                                 'min',
                                 'mk',
                                 'ml',
                                 'mn',
                                 'mo',
                                 'mr',
                                 'mrj',
                                 'ms',
                                 'mt',
                                 'mwl',
                                 'my',
                                 'myv',
                                 'mzn',
                                 'na',
                                 'nah',
                                 'nan',
                                 'nap',
                                 'nds',
                                 'ne',
                                 'new',
                                 'nl',
                                 'nn',
                                 'no',
                                 'nov',
                                 'nrm',
                                 'nso',
                                 'nv',
                                 'oc',
                                 'old',
                                 'om',
                                 'or',
                                 'os',
                                 'pa',
                                 'pag',
                                 'pam',
                                 'pap',
                                 'pcd',
                                 'pdc',
                                 'pfl',
                                 'pih',
                                 'pl',
                                 'pms',
                                 'pnb',
                                 'ps',
                                 'pt',
                                 'qu',
                                 'rm',
                                 'rn',
                                 'ro',
                                 'ru',
                     'rue',
                     'rw',
                     'sa',
                     'sah',
                     'sc',
                     'scn',
                     'sco',
                     'sd',
                     'se',
                     'sg',
                     'sh',
                     'si',
                     'sk',
                     'sl',
                     'sm',
                     'smn',
                     'sms',
                     'sn',
                     'so',
                     'sq',
                     'sr',
                     'srn',
                     'ss',
                     'stq',
                     'su',
                     'sv',
                     'sw',
                     'szl',
                     'ta',
                     'te',
                     'tet',
                     'tg',
                     'th',
                     'tk',
                     'tl',
                     'tn',
                     'to',
                     'tpi',
                     'tr',
                     'ts',
                     'tt',
                     'tw',
                     'ty',
                     'tzl',
                     'udm',
                     'ug',
                     'uk',
                     'ur',
                     'uz',
                     'vec',
                     'vep',
                     'vi',
                     'vls',
                     'vo',
                     'wa',
                     'war',
                     'wo',
                     'wuu',
                     'xal',
                     'xh',
                     'xmf',
                     'yi',
                     'yo',
                     'yue',
                     'za',
                     'zea',
                     'zh',
                     'zu'},
            'network': {'wikipedia', 'wikidata'},
            'nist': {'state_fips', 'fips_code'},
            'notable_tenant': {'wikidata'},
            'note': {'lanes', 'old_railway_operator'},
            'official_name': {'eo', 'nl', 'en', 'cs', 'din', 'vi', 'pl'},
            'old_name': {'eo', 'ru', 'en', 'vi', 'pl'},
            'old_short_name': {'ru'},
            'operator': {'wikipedia', 'type', 'short', 'wikidata'},
            'osmc': {'symbol'},
            'parking': {'fee'},
            'payment': {'american_express',
                        'apple_pay',
                        'bitcoin',
                        'cash',
                        'cheque',
                        'coins',
                        'contactless',
                        'credit_cards',
                        'debit_cards',
                        'discover_card',
                        'ebt',
                        'google_pay',
                        'litecoin',
                        'mastercard',
                        'notes',
                        'snap',
                        'visa',
                        'visa_debit',
                        'wic'},
            'phone': {'alt'},
            'plant': {'source', 'method'},
            'playground': {'theme'},
            'public_transport': {'version'},
            'radio_transponder': {'category', 'callsign'},
            'railway': {'ref', 'track_ref', 'position', 'historic', 'traffic_mode'},
            'ramp': {'wheelchair'},
            'ref': {'fips',
                    'hmdb',
                    'njhrp',
                    'njrhp',
                    'nrhp',
                    'penndot',
                    'retrofitness',
                    'walmart'},
            'restriction': {'hgv', 'conditional'},
            'roof': {'colour', 'shape', 'levels'},
            'short_name': {'nl', 'en', 'ru', 'be', 'zh', 'cs', 'vi', 'es', 'mo', 'pl'},
            'social_facility': {'for'},
            'socket': {'chademo', 'tesla_supercharger'},
            'source': {'alt_name',
                       'cycleway',
                       'date',
                       'geometry',
                       'lanes',
                       'maxspeed',
                       'name',
                       'note',
                       'old_ref',
                       'oneway',
                       'population',
                       'short_name',
                       'surface',
                       'unsigned_ref'},
            'swing_gate': {'type'},
            'theatre': {'type'},
            'tiger': {'cfcc',
                      'county',
                      'name_base',
                      'name_direction_prefix',
                      'name_direction_suffix',
                      'name_type',
                      'reviewed',
                      'separated',
                      'source',
                      'tlid',
                      'upload_uuid',
                      'zip_left',
                      'zip_right'},
            'toilets': {'disposal', 'wheelchair'},
            'tower': {'construction', 'type'},
            'traffic_signals': {'direction'},
            'turn': {'lanes'},
            'was': {'amenity'},
            'website': {'state'}}
```

Looks like there are some wierd name tags in different languages.



```
<tag k="name" v="Trenton"/>
<tag k="name:en" v="Trenton"/>
<tag k="name:pl" v="Trenton"/>
<tag k="name:ru" v="Трентон"/>
<tag k="name:ta" v="இட்ரென்டன்"/>
<tag k="name:uk" v="Трентон"/>
<tag k="name:zh" v="特伦顿 / 翠登 / 特倫頓"/>
<tag k="name:zh-Hans" v="特伦顿"/>
<tag k="name:zh-Hant" v="翠登 / 特倫頓"/>
<tag k="name:zh-hk" v="特倫頓"/>
<tag k="name:zh-tw" v="翠登"/>
```

One of the tag that starts with brand has a subtype as 'wikipedia'. Will change it to 'Type'.

In [106]:
```
import xml.etree.ElementTree as ET
from collections import defaultdict
import pprint

def process_map(filename):
    brands = defaultdict(set)
    for event, elem in ET.iterparse(mapfile, events=("start",)):

        if elem.tag == "node" or elem.tag == "way":
            for tag in elem.iter("tag"):
                k = tag.attrib['k']
                v = tag.attrib['v']
                if k.startswith('brand'):
                    ka = k.split(':')  # create a list of k-values, split on :
(creates array of len 1 if no :)
                    if len(ka) == 1:  # wasn't colon-separated, so was a "typ
e" of lift. Create "type" of lift
                        ka.append('type')
                    if ka[1] == 'wikipedia': # bad tag -- ignore it
                        continue
                    else:
                        brands[ka[1]].add(v)
    return brands


with open('trenton-princeton.xml', 'rb') as mapfile:
    brands = process_map(mapfile)
    pprint.pprint(brands)
```

```
            defaultdict(<class 'set'>,
                        {'type': {'7-Eleven',
                                  'ALDI',
                                  'AMC',
                                  'AT&T',
                                  'Acme',
                                  'Advance Auto Parts',
                                  'Alex and Ani',
                                  'American Automobile Association',
                                  'Ann Taylor',
                                  "Applebee's Neighborhood Grill & Bar",
                                  'AutoZone',
                                  "BJ's Wholesale Club",
                                  'BP',
                                  'Banana Republic',
                                  'Bank of America',
                                  'Barnes & Noble',
                                  'Baskin-Robbins',
                                  'Bed Bath & Beyond',
                                  'Benefit',
                                  'Best Buy',
                                  'Best Western',
                                  'Blimpie',
                                  "Bob's Discount Furniture",
                                  'Bonefish Grill',
                                  'Boost Mobile',
                                  'Boston Market',
                                  'Boys & Girls Club',
                                  'Brooks Brothers',
                                  'Buffalo Wild Wings',
                                  'Burger King',
                                  'Buy Buy Baby',
                                  'CVS',
                                  'CVS Pharmacy',
                                  'Chase',
                                  'Chevrolet',
                                  'Chick-fil-A',
                                  "Chili's",
                                  'Chipotle',
                                  "Chuck E. Cheese's",
                                  'Citgo',
                                  'Clarion',
                                  'Coldwell Banker',
                                  'Conoco',
                                  'Costco',
                                  'Costco Gasoline',
                                  'Courtyard',
                                  'Cracker Barrel',
                                  'DSW',
                                  'Dairy Queen',
                                  "David's Bridal",
                                  'Delta',
                                  'Dero',
                                  "Dick's Sporting Goods",
                                  'Dollar General',
                                  'Dollar Tree',
                                  "Domino's",
 "Domino's Pizza",
 'DoubleTree',
 'Dress Barn',
 "Dunkin'",
 "Dunkin' Donuts",
 'Enterprise',
 'Extended Stay America',
 'Exxon',
 'Family Dollar',
 'Famous Footwear',
 'FedEx Office',
 'Five Below',
 'Five Guys',
 'GMC',
 'GameStop',
 'Gap',
 'Getty',
 'Goodwill',
 'Goodyear',
 'Gulf',
 'H&R Block',
 'Hampton',
 'Hand & Stone Massage and Facial Spa',
 'Harbor Freight Tools',
 'Hobby Lobby',
 'Holiday Inn',
 'HomeGoods',
 'Homewood Suites',
 'Hooters',
 'Hyatt Place',
 'Hyundai',
 'IHOP',
 'Investors Bank',
 'J.Crew',
 'Jackson Hewitt',
 'Jiffy Lube',
 'KFC',
 'KinderCare',
 "Kohl's",
 'Liberty Tax',
 'Life Storage',
 'Little Caesars',
 'LongHorn Steakhouse',
 "Lowe's",
 'Lukoil',
 'Lululemon',
 'MAC Cosmetics',
 'Marshalls',
 'Mattress Firm',
 'Mazda',
 "McDonald's",
 "Men's Wearhouse",
 'MetroPCS',
 'Michaels',
 'Midas',
 "Mimi's Cafe",
 "Modell's Sporting Goods",
 'Monro Muffler Brake',
 'Nissan',
 'Nordstrom Rack',
 'Old Navy',
 'Olive Garden',
 'On The Border',
 'Origins',
 'Outback Steakhouse',
 "P.F. Chang's",
 'PNC Bank',
 'Panera Bread',
 'Paper Source',
 'Party City',
 'Penske Truck Rental',
 'Pep Boys',
 'Perkins',
 'PetSmart',
 'Phillips 66',
 'Pier 1 Imports',
 'Pioneer',
 'Pizza Hut',
 'Planned Parenthood',
 'Pollo Campero',
 'Popeyes',
 'Porsche',
 'Qdoba',
 'Quality Inn',
 'QuickChek',
 'REI',
 'Rainbow',
 'Raymour & Flanigan',
 'Red Lobster',
 'Red Robin',
 'Red Roof Inn',
 'Regal Cinemas',
 'Rent-A-Center',
 'Residence Inn',
 'Retro Fitness',
 "Rita's Italian Ice",
 'Rite Aid',
 "Romano's Macaroni Grill",
 'Ross',
 'Santander',
 'Shell',
 'ShopRite',
 'Snipes',
 'Sonesta',
 'Sonic',
 'Sprint',
 'Staples',
 'Starbucks',
 'State Farm',
 'Stop & Shop',
 'Subway',
 'Sunoco',
 'Supercuts',
 'T-Mobile',
                                          'TD Bank',
                                          "TGI Friday's",
                                          'TJ Maxx',
                                          'Taco Bell',
                                          'Talbots',
                                          'Target',
                                          'Tesla Supercharger',
                                          'The Goddard School',
                                          'The Home Depot',
                                          'The UPS Store',
                                          'Toyota',
                                          'Tractor Supply Company',
                                          "Trader Joe's",
                                          'U-Haul',
                                          'US Post Office',
                                          'Ulta Beauty',
                                          'United States Post Office',
                                          'Urban Outfitters',
                                          'Valero',
                                          'Verizon Wireless',
                                          'Volkswagen',
                                          'Walgreens',
                                          'Walmart',
                                          'Wawa',
                                          'Wegmans',
                                          'Wells Fargo',
                                          "Wendy's",
                                          'Westin',
                                          'Whole Foods Market',
                                          'YMCA',
                                          'Zoës Kitchen',
                                          'eVgo',
                                          'gmc;buick;pontiac'},
                                 'wikidata': {'Q1025921',
                                              'Q1046951',
                                              'Q1053170',
                                              'Q1064893',
                                              'Q1072948',
                                              'Q1075788',
                                              'Q10860683',
                                              'Q11288478',
                                              'Q1131810',
                                              'Q1141226',
                                              'Q1185675',
                                              'Q1283291',
                                              'Q1296860',
                                              'Q1330910',
                                              'Q1363885',
                                              'Q1373493',
                                              'Q1393809',
                                              'Q1423218',
                                              'Q145168',
                                              'Q15109896',
                                              'Q15109925',
                                              'Q152057',
                                              'Q1524184',
                                              'Q154950',
                                              'Q157169',
                                              'Q15903261',
                                              'Q1591889',
                                              'Q16198810',
                                              'Q1656230',
                                              'Q17022081',
                                              'Q17085454',
                                              'Q17089620',
                                              'Q177054',
                                              'Q1809448',
                                              'Q191615',
                                              'Q1925685',
                                              'Q1969162',
                                              'Q2007336',
                                              'Q202210',
                                              'Q2078880',
                                              'Q21463374',
                                              'Q214763',
                                              'Q2438391',
                                              'Q244457',
                                              'Q246',
                                              'Q25000269',
                                              'Q2504643',
                                              'Q2553262',
                                              'Q259340',
                                              'Q2624442',
                                              'Q2717882',
                                              'Q2735242',
                                              'Q2895769',
                                              'Q28993',
                                              'Q2923055',
                                              'Q294721',
                                              'Q29570',
                                              'Q298594',
                                              'Q301965',
                                              'Q3045312',
                                              'Q3259007',
                                              'Q327634',
                                              'Q329347',
                                              'Q3307147',
                                              'Q3312613',
                                              'Q3317844',
                                              'Q3375007',
                                              'Q3414933',
                                              'Q341975',
                                              'Q3433273',
                                              'Q3442791',
                                              'Q3552193',
                                              'Q35996',
                                              'Q3658429',
                                              'Q37158',
                                              'Q38076',
                                              'Q38928',
                                              'Q40993',
                                              'Q41171672',
                                              'Q420822',
                                              'Q42306166',
                                              'Q4492609',
                                              'Q463436',
                                              'Q465751',
                                              'Q4686051',
                                              'Q474379',
                                              'Q4766699',
                                              'Q4781944',
                                              'Q4826087',
                                              'Q483551',
                                              'Q4835754',
                                              'Q487907',
                                              'Q491516',
                                              'Q4926479',
                                              'Q4931582',
                                              'Q4941599',
                                              'Q4943790',
                                              'Q5003352',
                                              'Q509255',
                                              'Q5206207',
                                              'Q5230388',
                                              'Q524629',
                                              'Q524757',
                                              'Q5272601',
                                              'Q5289230',
                                              'Q53268',
                                              'Q533415',
                                              'Q5360181',
                                              'Q5370765',
                                              'Q5421850',
                                              'Q5433101',
                                              'Q5433457',
                                              'Q5455836',
                                              'Q550258',
                                              'Q55613342',
                                              'Q5576260',
                                              'Q5583655',
                                              'Q5627799',
                                              'Q5646230',
                                              'Q5654601',
                                              'Q57405513',
                                              'Q5835668',
                                              'Q584601',
                                              'Q5874938',
                                              'Q5887941',
                                              'Q5890701',
                                              'Q5936320',
                                              'Q603617',
                                              'Q6117132',
                                              'Q61803820',
                                              'Q6192247',
                                              'Q61994955',
                                              'Q620875',
                                              'Q621532',
                                              'Q62274661',
                                              'Q6410551',
                                              'Q65090033',
                                              'Q6541978',
                                              'Q6643229',
                                              'Q668687',
                                              'Q6702957',
                                              'Q6791878',
                                              'Q6835667',
                                              'Q688825',
                                              'Q6902090',
                                              'Q7091305',
                                              'Q7130852',
                                              'Q7140896',
                                              'Q715583',
                                              'Q7169056',
                                              'Q7191691',
                                              'Q72629292',
                                              'Q7271689',
                                              'Q7284708',
                                              'Q7299290',
                                              'Q7304886',
                                              'Q7304949',
                                              'Q7313497',
                                              'Q7315394',
                                              'Q7336456',
                                              'Q7362714',
                                              'Q738853',
                                              'Q744149',
                                              'Q7501097',
                                              'Q752941',
                                              'Q7561808',
                                              'Q7643239',
                                              'Q7669891',
                                              'Q7679064',
                                              'Q7771029',
                                              'Q78165540',
                                              'Q785943',
                                              'Q7862902',
                                              'Q7880076',
                                              'Q7882701',
                                              'Q795454',
                                              'Q806085',
                                              'Q8074747',
                                              'Q81234570',
                                              'Q813782',
                                              'Q830334',
                                              'Q835638',
                                              'Q839466',
                                              'Q846301',
                                              'Q847743',
                                              'Q864407',
                                              'Q919641',
                                              'Q929722',
                                              'Q942741',
                                              'Q967265'}})
```

Check Unique User Ids and Names

In [107]:
```python
import xml.etree.ElementTree as ET
import pprint
import re

def get_userid(element):
    return element.attrib['uid']

def get_username(element):
    return element.attrib['user']

def process_map(filename):
    userids = set()
    usernames = set()
    for _, element in ET.iterparse(filename):
        if 'uid' in element.attrib.keys():
            userid = get_userid(element)
            userids.add(userid)
        if 'user' in element.attrib.keys():
            username = get_username(element)
            usernames.add(username)

    return userids, usernames

with open('trenton-princeton.xml', 'rb') as mapfile:
    userids, usernames = process_map(mapfile)
    print ("There are %d unique user IDs" % len(userids))
    print ("There are %d unique usernames" % len(usernames))
```

```
There are 1458 unique user IDs
There are 1458 unique usernames
```

Auditing Street Types

Find if there are any possible street type problems

In [108]:
```
import xml.etree.cElementTree as ET
from collections import defaultdict
import re
import pprint

street_type_re = re.compile(r'\b\S+\.?$', re.IGNORECASE)


expectedTypes = ["Street", "Avenue", "Boulevard", "Drive", "Court", "Circle",
"Road"]


def audit_street_type(street_types, street_name, street_types_count):
    m = street_type_re.search(street_name)
    if m:
        street_type = m.group()
        if street_type not in street_types_count.keys():
            street_types_count[street_type] = 1
        else:
            street_types_count[street_type] += 1
        if street_type not in expectedTypes:
            street_types[street_type].add(street_name)

    return street_types_count

def is_street_name(elem):
    return (elem.attrib['k'] == "addr:street")

def audit(mapfile):
    street_types = defaultdict(set)
    street_types_count = {}
    for event, elem in ET.iterparse(mapfile, events=("start",)):

        if elem.tag == "node" or elem.tag == "way":
            for tag in elem.iter("tag"):
                if is_street_name(tag):
                    street_types_count = audit_street_type(street_types, tag.a
ttrib['v'], street_types_count)

    return street_types, street_types_count

with open('trenton-princeton.xml', 'rb') as mapfile:
    street_types, street_types_count = audit(mapfile)
    print ("Find if there are any street type problems:")
    pprint.pprint(street_types)
    print ("Count of Street Types:")
    pprint.pprint(street_types_count)
```

```
Find if there are any street type problems:
defaultdict(<class 'set'>,
            {'1': {'US Highway 1', 'Route 1'},
             '111': {'South Clinton Avenue Ste. 111'},
             '130': {'Route 130',
                     'U.S. 130',
                     'US 130',
                     'US Highway 130',
                     'US Hwy 130',
                     'US Route 130',
                     'Us Highway 130'},
             '160': {'Carnegie Center West, Suite 160'},
             '19067': {'East Trenton Avenue Morrisville, PA 19067'},
             '27': {'Route 27'},
             '29': {'Route 29'},
             '31': {'NJ 31', 'Rt 31'},
             '33': {'Highway 33',
                    'NJ 33',
                    'New Jersey State Highway 33',
                    'Route 33',
                    'Rte 33',
                    'State Highway 33'},
             '683': {'County Road 683'},
             'Ave': {'1655 North Olden Ave',
                     'Alden Ave',
                     'N Olden Ave',
                     'Parkway Ave',
                     'Pennington Ave',
                     'S Clinton Ave',
                     'S. Clinton Ave',
                     'Sloan Ave',
                     'West Trenton Ave',
                     'Yard Ave'},
             'Ave.': {'Broad Ave.'},
             'B': {'Princeton Ave #B'},
             'Blvd': {'Floral Vale Blvd',
                      'Nassau Park Blvd',
                      'Nassau Parl Blvd',
                      'Route 1 South and Promenade Blvd',
                      'Summerfield Blvd'},
             'Center': {'Overlook Center'},
             'Cir': {'Blossom Cir'},
             'Cres': {'Ely Cres'},
             'Crossing': {"Washington's Crossing"},
             'Ct': {'Rousillon Ct', 'Heller Park Ct'},
             'Dr': {'Clubhouse Dr',
                    'Lawrence Dr',
                    'Merrill Lynch Dr',
                    'Old Stone Mill Dr',
                    'Wood Mill Dr',
                    'Wyncrest Dr'},
             'E': {'College Rd E'},
             'East': {'460 Route 33 East',
                      'College Road East',
                      'Palmer Square East'},
             'Extension': {'Spruce Street Extension'},
             'H': {'Robbinsville Allentown Rd Ste H'},
             'Harrison': {'Harrison'},
             'Highway': {'Lincoln Highway'},
             'Lane': {'Abbington Lane',
                      'Anderson Lane',
                      'Arnold Lane',
                      'Balsam Lane',
                      'Cedar Lane',
                      'Chicory Lane',
                      'Craven Lane',
                      'Darrah Lane',
                      'East Lincoln Lane',
                      'Eddington Lane',
                      'Green Lane',
                      'Hornor Lane',
                      'Independence Lane',
                      'Inverness Lane',
                      'Kingston Lane',
                      'Kingstone Lane',
                      'Leavitt Lane',
                      'Leshin Lane',
                      'Linden Lane',
                      'Lockewood Lane',
                      'Palmer Lane',
                      'Pemberton Lane',
                      'Reagan Lane',
                      'Shadowstone Lane',
                      'Snowden Lane',
                      'South Olden Lane',
                      'Stouts Lane',
                      'Sussex Lane'},
             'Ln': {'Olden Ln'},
             'N': {'Route 31 N', 'US-1 N', 'US 130 N'},
             'NJ-31': {'NJ-31'},
             'NJ-33': {'NJ-33'},
             'North': {'Greenfield Drive North'},
             'PIke': {'Princeton PIke'},
             'Pike': {'Brunswick Pike', 'Princeton Pike'},
             'Place': {'Berkley Place',
                       'Berwyn Place',
                       'Broemel Place',
                       'Chancery Place',
                       'Edwards Place',
                       'Evelyn Place',
                       'Halifax Place',
                       'Hamilton Health Place',
                       'Library Place',
                       'Loetscher Place',
                       'Montgomery Place',
                       'Murray Place',
                       'Park Place',
                       'Queenston Place',
                       'Tee-Ar Place',
                       'University Place',
                       'Warwick Place'},
             'Plaza': {'Riverview Plaza'},
             'Rd': {'Campus Rd',
                    'Cranbury Rd',
                    'Ridge Rd',
                    'Robbinsville Allentown Rd',
                    'Robbinsville-Allentown Rd',
                    'Schalks Crossing Rd',
                    'Whitehorse Hamilton Square Rd'},
             'Rd.': {'Lawrence Rd.'},
             'Robbinsvil': {'Allentown Robbinsvil'},
             'S': {'US-1 S'},
             'South': {'US 1 South', 'Forrestal Road South'},
             'Square': {'North Commerce Square'},
             'St': {'Alexander St',
                    'Cooper St',
                    'E State St',
                    'Market St',
                    'N Broad St',
                    'N Main St',
                    'Nassau St',
                    'S Broad St'},
             'Terrace': {'Lakeview Terrace', 'Queensboro Terrace'},
             'US-1': {'US-1'},
             'US-130': {'US-130'},
             'W': {'International Road W'},
             'Way': {'Allerton Way',
                     'Azalea Way',
                     'Barto Way',
                     'Beethoven Way',
                     'Bristol Way',
                     'Brookfield Way',
                     'Capital Way',
                     'Carriage Way',
                     'Chelten Way',
                     'Durham Way',
                     'Fulham Way',
                     'Hickory Way',
                     'Independence Way',
                     'John Fitch Way',
                     'Justice Samuel A Alito Jr Way',
                     'New Canton Way',
                     'Nottingham Way',
                     'Petal Way',
                     'Rembrandt Way',
                     'Roosevelt Way',
                     'Sculptors Way',
                     'Southern Way',
                     'Spencer Way',
                     'Strathmore Way',
                     'Sturges Way',
                     'Sullivan Way',
                     'Sylvia Way',
                     'Walt Whitman Way',
                     'West Manor Way',
                     'Western Way',
                     'Windward Way'},
             'West': {'College Road West', 'Carnegie Center West'},
             'al': {'pullen al'},
             'ave': {'Stuart ave',
                     'and 1891 brunswick ave',
                     'hillcrest ave',
                     'michigan ave'},
             'avenue': {'Ohio avenue'},
             'dr': {'hampshire dr'},
             'ext': {'brunswick ave ext'},
             'rd': {'Lawrenceville rd'},
             'st': {'centre st',
                    'clay st',
                    'e front st',
                    'e state st',
                    'jackson st',
                    'livingston st',
                    'market st',
                    'mercer st',
                    'pear st',
                    's broad st',
                    's montgomery st',
                    's stockton st',
                    's warren st',
                    'st',
                    'w state st'}})
Count of Street Types:
{'1': 3,
 '111': 1,
 '130': 9,
 '160': 1,
 '19067': 1,
 '27': 3,
 '29': 1,
 '31': 2,
 '33': 19,
 '683': 3,
 'Ave': 12,
 'Ave.': 1,
 'Avenue': 2020,
 'B': 1,
 'Blvd': 5,
 'Boulevard': 153,
 'Center': 1,
 'Cir': 1,
 'Circle': 88,
 'Court': 417,
 'Cres': 2,
 'Crossing': 1,
 'Ct': 2,
 'Dr': 37,
 'Drive': 887,
 'E': 1,
 'East': 12,
 'Extension': 44,
 'H': 1,
 'Harrison': 1,
 'Highway': 3,
 'Lane': 451,
 'Ln': 1,
 'N': 3,
 'NJ-31': 1,
 'NJ-33': 3,
 'North': 1,
 'PIke': 2,
 'Pike': 343,
 'Place': 94,
 'Plaza': 4,
 'Rd': 10,
 'Rd.': 1,
 'Road': 1525,
 'Robbinsvil': 1,
 'S': 2,
 'South': 2,
 'Square': 1,
 'St': 11,
 'Street': 1772,
 'Terrace': 25,
 'US-1': 1,
 'US-130': 2,
 'W': 2,
 'Way': 308,
 'West': 2,
 'al': 3,
 'ave': 25,
 'avenue': 192,
 'dr': 1,
 'ext': 1,
 'rd': 11,
 'st': 458}
```

From above list duplicate entries of street types are seen as under:

1. -2 entries for 'cir' and 88 for 'circle'. Will change 'cir' to circle.
2. -2 entries for 'S' and 2 for 'south'. Will change 'S' to South.
3. -2 entries for 'W' and 308 for 'way'. Will change w to way.
4. -5 entries for 'blvd' and 153 for 'Boulevard'. Will change blvd to Boulevard.
5. -Similarly there are duplicate enrtries for Road, Street, Court, avenue.
6. -City has multiple entries of township as 'twp & TWP'. This need to be changed to township.
7. -ave tag has a wrong name:'and 1891 brunswick avenue'.This need to be corrected to 'Brunswick avenue'.

First clean up data and then save into JSON file format.¶

In [109]:
```
import xml.etree.ElementTree as ET
import pprint
import re
import codecs
import json


lower = re.compile(r'^([a-z]|_)*$')
lower_colon = re.compile(r'^([a-z]|_)*:([a-z]|_)*$')
problemchars = re.compile(r'[=\+/&<>;\'"\?%#$@\,\. \t\r\n]')

CREATED = [ "version", "changeset", "timestamp", "user", "uid"]


def shape_element(element):
    node = {}
    pos = ['lat', 'lon']
    created = ['version', 'changeset', 'timestamp', 'user', 'uid']
    if element.tag == "node" or element.tag == "way" :

        node['created'] = {}
        node['type'] = element.tag
        for attrib, value in element.attrib.items():
            if attrib in pos:
                if 'pos' not in node.keys():
                    node['pos'] = [0,0]  # only create position sub-document i
f position is one of the attributes
                if attrib == 'lat':
                    node['pos'][0] = float(value)
                else:
                    node['pos'][1] = float(value)
            elif attrib in created:
                node['created'][attrib] = value
            else:
                node[attrib] = value


        for tag in element.iter("tag"):
            k = tag.attrib['k']
            v = tag.attrib['v']
            if k.startswith('addr'): # Format addresses into subdocument
                addrtag = k.split(':')
                if len(addrtag) > 2:
                    continue
                if 'address' not in node.keys():
                    node['address'] = {}
                node['address'][addrtag[1]] = v
            elif k.startswith('brand'):  # Format brand into types
                atag = k.split(':')  # create a list of k-values, split on :
                if len(atag) == 1:
                    atag.append('type')
                if atag[1] == 'wikipedia': # bad tag -- ignore it
                    continue
                if 'brand' not in node.keys():
                    node['brand'] = {}
                node['brand'][atag[1]] = v
            #below code is to update the street abbreviated names like Rd., C
T, DR ,blvd to full names.
                    #to full name to make sure data is in sync.
            elif k.startswith('addr:street'):
                ptag = k.split(':')
                if len(ptag) == 1:
                    ptag.append('street')
                    if ptag.endswith(' Rd') or ptag.endswith(' rd') or ptag.en
dswith(' Rd.'):
                        ptag.replace('Rd', 'Road')
                        ptag.replace('Rd.', 'Road')
                        ptag.replace('rd', 'Road')
                        ptag.replace('CT', 'Court')
                        ptag.replace('DR.', 'Drive')
                        ptag.replace('blvd','Boulevard')
                        ptag.replace('cir','Circle')
                        ptag.replace('S','South')
                        ptag.replace(' st','Street')
                        ptag.replace('PIke','Pike')
                        ptag.replace('and 1891 brunswick avenue','brunswick av
enue')
                        ptag.replace('and 1891 brunswick ave','brunswick avenu
e')

            elif k.startswith('addr:'): # Format adress:street into sub-docume
nt
                ptag = k.split(':')
                if len(ptag) == 1:
                    ptag.append('City') #City tag has township in different fo
rmats. Updating so all sync as one nanme.
                    if ptag.endswith(' twp') or ptag.endswith(' Twp'):
                        ptag.replace('twp', 'Township')
                        ptag.replace('Twp', 'Township')
                elif k.startswith('tiger'): # Format tiger tags into sub-docum
ent
                    ttag = k.split(':')
                if len(ttag) == 1:
                    ptag.append('misc')
                if 'tiger' not in node.keys():
                    node['tiger'] = {}
                node['tiger'][ttag[1]] = v
            else:
                node[k] = v

        for tag in element.iter("nd"):
            if 'node_refs' not in node.keys():
                node['node_refs'] = []
            ref = tag.attrib['ref']
            node['node_refs'].append(ref)

    return node

def process_map(file_in, pretty = False):
    file_out = "Trenton_Princeton_Maps.json".format(file_in)
    data = []
    with codecs.open(file_out, "w") as fo:
        for _, element in ET.iterparse(file_in):
            el = shape_element(element)
            if el:
                data.append(el)
                if pretty:
                    fo.write(json.dumps(el, indent=2)+"\n")
                else:
                    fo.write(json.dumps(el) + "\n")
    return data



with open('Trenton-Princeton.xml', 'rb') as mapfile:
    data_array = process_map(mapfile, False)
```

Auditing ends here. Data is loaded into JSON. Rest of the data cleanup will be done in Mongo DB.

MongoDB

Open Connection to MongoDB

In [114]:
```
#import modules
import pymongo
from pymongo import MongoClient
import pprint

#Open Connection
client = MongoClient('mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000')
```

Open Database

In [115]:
```
#Open Database
db = client.db_Trenton_Princeton
db = client['db_Trenton_Princeton']
print(db)
```

```
Database(MongoClient(host=['127.0.0.1:27017'], document_class=dict, tz_aware=
False, connect=True, directconnection=True, serverselectiontimeoutms=2000),
'db_Trenton_Princeton')
```

Print Collections

In [116]:
```
#Print Collections
collections = db.list_collection_names()
print ("collections:", collections, "\n")
```

```
collections: ['TrentPrincemap']
```

Find Nodes

In [117]:
```
#Find Nodes
nodes=db.TrentPrincemap.count_documents({'type':'node'})
pprint.pprint("count of nodes")
pprint.pprint(nodes)
```

```
'count of nodes'
686748
```

Find Ways

In [118]:
```
#Find Ways
ways=db.TrentPrincemap.count_documents({'type':'way'})
pprint.pprint("count of ways")
pprint.pprint(ways)
```

```
'count of ways'
80210
```

Find unique Users

In [119]:
```
u=len(db.TrentPrincemap.distinct('created.user'))
pprint.pprint("Number of Unique Users")
pprint.pprint(u)
```

```
'Number of Unique Users'
1437
```

Top 5 Users

In [120]:
```
topusers  = db.TrentPrincemap.aggregate([
    {"$group" :
        {"_id" : "$created.user",
         "count" : {"$sum" : 1}}},
          { '$sort' : { 'count' : -1 } }, { '$limit' : 5 }
                              ])
for k in topusers:
    print(k)
```

```
{'_id': 'RM23', 'count': 91635}
{'_id': 'woodpeck_fixbot', 'count': 68305}
{'_id': 'NJDataUploads', 'count': 44499}
{'_id': 'Utible', 'count': 26643}
{'_id': 'James Michael DuPont', 'count': 26480}
```

Find Amenities & Highways

In [121]:
```
#Find Service Highways
s=db.TrentPrincemap.count_documents({"highway": "service"})
pprint.pprint("count of services")
pprint.pprint(s)

#Find Residential Highways
r=db.TrentPrincemap.count_documents({"highway": "residential"})
pprint.pprint("count of residential areas")
pprint.pprint(r)

#Find number of parking sites
p=db.TrentPrincemap.count_documents({"amenity": "parking"})
pprint.pprint("count of parking areas")
pprint.pprint(p)
```

```
'count of services'
22319
'count of residential areas'
9314
'count of parking areas'
1978
```

List number of shops

In [128]:
```
shops  = db.TrentPrincemap.aggregate([
    {"$group" :
        {"_id" : "$shop",
         "count" : {"$sum" : 1}}},
          { '$sort' : { 'count' : -1 } }, { '$limit' : 25 }
                              ])
for sh in shops:
    print(sh)
```

```
{'_id': None, 'count': 765905}
{'_id': 'convenience', 'count': 134}
{'_id': 'hairdresser', 'count': 124}
{'_id': 'car_repair', 'count': 97}
{'_id': 'clothes', 'count': 54}
{'_id': 'deli', 'count': 51}
{'_id': 'alcohol', 'count': 49}
{'_id': 'car', 'count': 46}
{'_id': 'supermarket', 'count': 41}
{'_id': 'yes', 'count': 28}
{'_id': 'variety_store', 'count': 27}
{'_id': 'mobile_phone', 'count': 27}
{'_id': 'beauty', 'count': 26}
{'_id': 'laundry', 'count': 24}
{'_id': 'funeral_directors', 'count': 16}
{'_id': 'jewelry', 'count': 15}
{'_id': 'bakery', 'count': 15}
{'_id': 'furniture', 'count': 14}
{'_id': 'department_store', 'count': 12}
{'_id': 'mall', 'count': 11}
{'_id': 'dry_cleaning', 'count': 11}
{'_id': 'books', 'count': 11}
{'_id': 'gift', 'count': 10}
{'_id': 'storage_rental', 'count': 10}
{'_id': 'tyres', 'count': 9}
```

From the top 25 shops output derived from above code, it can be noticed that there are some counts for 'yes' as a value of shop instead of type. Let's correct 'yes' into meaningful type.

Check how many shops have 'yes'

In [129]:
```
m=db.TrentPrincemap.find({"shop": "yes"}, {"name":1});
num=0
for p in m:
    pprint.pprint(p)
    num += 1
```

```
{'_id': ObjectId('61244af781d601a4ad6b5ffe'), 'name': 'Carter & Cavero'}
{'_id': ObjectId('61244b0681d601a4ad6f2fcf'),
 'name': 'M. Temkin Store Fixtures'}
{'_id': ObjectId('61244b0681d601a4ad6f3146'), 'name': 'Bori Cuba'}
{'_id': ObjectId('61244b0681d601a4ad6f3160'), 'name': 'Home Rubber Co.'}
{'_id': ObjectId('61244b0681d601a4ad6f331f'), 'name': 'His Music Her Hands'}
{'_id': ObjectId('61244b0681d601a4ad6f33a5'), 'name': 'Botanica Las 2 Podere
s'}
{'_id': ObjectId('61244b0681d601a4ad6f3a9a'), 'name': 'Solar Home Brew'}
{'_id': ObjectId('61244b0681d601a4ad6f412e'), 'name': 'Muirhead Foods'}
{'_id': ObjectId('61244b0681d601a4ad6f41b4'), 'name': 'HomeLife Company'}
{'_id': ObjectId('61244b0781d601a4ad6f41f5'), 'name': 'ARSI Liquidators'}
{'_id': ObjectId('61244b0781d601a4ad6f4206'), 'name': 'Titan Rack & Shelvin
g'}
{'_id': ObjectId('61244b0781d601a4ad6f42e2'),
 'name': 'J.W. Kennedy Fire Protection Services'}
{'_id': ObjectId('61244b0781d601a4ad6f4894'), 'name': 'Capitol Fire Protectio
n'}
{'_id': ObjectId('61244b0781d601a4ad6f48f1'), 'name': 'Van Syckel House'}
{'_id': ObjectId('61244b0781d601a4ad6f4a2e'), 'name': 'The Costume Scene'}
{'_id': ObjectId('61244b0781d601a4ad6f4a8c'), 'name': "Case's Pork Roll"}
{'_id': ObjectId('61244b0781d601a4ad6f4a91'),
 'name': "Porfirio's Italian Foods"}
{'_id': ObjectId('61244b0781d601a4ad6f4ad9'), 'name': 'Nimba Food Store'}
{'_id': ObjectId('61244b0781d601a4ad6f4ade'), 'name': 'Internova Multiservice
s'}
{'_id': ObjectId('61244b0781d601a4ad6f4b5a'),
 'name': 'Creative Arts Trophy Shop'}
{'_id': ObjectId('61244b0781d601a4ad6f4b85'), 'name': 'Tipicos La Quetzaltec
a'}
{'_id': ObjectId('61244b0781d601a4ad6f4b93'), 'name': 'Botanica Santa Marth
a'}
{'_id': ObjectId('61244b0781d601a4ad6f4bab'), 'name': 'G.E. Marshall'}
{'_id': ObjectId('61244b0781d601a4ad6f4c18'), 'name': 'Prince Multi Service
s'}
{'_id': ObjectId('61244b0781d601a4ad6f4c2a'), 'name': 'Banrural Envios Expres
s'}
{'_id': ObjectId('61244b0781d601a4ad6f4c34'), 'name': 'Salud y Figura'}
{'_id': ObjectId('61244b0781d601a4ad6f4c58'), 'name': 'Herrera Multiservice'}
{'_id': ObjectId('61244b0781d601a4ad6f4ece'), 'name': 'Snack King Vending'}
```

Let's fix by updating the shop names into different categories that has 'yes' as a value.

In [130]:
```
# Fixing the bad data healthcare issue
print("Lets first add selected shops by name into different categories")

# Update the healthcare in the document:use function update_many
coll=db.TrentPrincemap
updateResult = coll.update_one({'shop': 'yes','name':'J.W. Kennedy Fire Protec
tion Services'}, {'$set': {'shop': 'Fire Services'}})
updateResult2 = coll.update_one({'shop': 'yes','name':'Capitol Fire Protectio
n'}, {'$set': {'shop': 'Fire Services'}})
updateResult3 = coll.update_one({'shop': 'yes','name':'Herrera Multiservice'},
{'$set': {'shop': 'Multi Services'}})
updateResult4 = coll.update_one({'shop': 'yes','name':'Prince Multi Services'
}, {'$set': {'shop': 'Multi Services'}})
updateResult5 = coll.update_one({'shop': 'yes','name':'Internova Multiservice
s'}, {'$set': {'shop': 'Multi Services'}})
```

```
Lets first add selected shops by name into different categories
```

Now lets add all remaining shops into department store category. Note: Function update_many is used here to update in bulk.

In [131]:
```
# Update the healthcare in the document:use function update_many
coll=db.TrentPrincemap
updateResult = coll.update_many({'shop': 'yes'}, {'$set': {'shop': 'department
_store'}})
```

Check if any more 'yes' is left.

In [132]:
```
#----Check if 'shop': 'yes' is removed from the collection. This should give N
ONE.
x=db.TrentPrincemap.count_documents({ 'shop': 'yes'})
print(x)
```

```
0
```

In [133]:
```
#Count total number of amenities grouped by each amenity and display in  desce
nding sorting order.
#****************************
print('Total Number of Amenities from highest to lowest counts')
from bson.son import SON
s=db.TrentPrincemap.aggregate([
    {"$group" :
        {"_id" : "$amenity",
         "count" : {"$sum" : 1}
         }},  { '$sort' : { 'count' : -1 } }, { '$limit' : 20 }

])
for k in s:
    print(k)
```

```
Total Number of Amenities from highest to lowest counts
{'_id': None, 'count': 762509}
{'_id': 'parking', 'count': 1978}
{'_id': 'parking_space', 'count': 512}
{'_id': 'restaurant', 'count': 325}
{'_id': 'place_of_worship', 'count': 273}
{'_id': 'school', 'count': 221}
{'_id': 'fast_food', 'count': 113}
{'_id': 'fuel', 'count': 99}
{'_id': 'bench', 'count': 99}
{'_id': 'bank', 'count': 86}
{'_id': 'bar', 'count': 50}
{'_id': 'fountain', 'count': 48}
{'_id': 'fire_station', 'count': 42}
{'_id': 'shelter', 'count': 42}
{'_id': 'grave_yard', 'count': 39}
{'_id': 'cafe', 'count': 34}
{'_id': 'pharmacy', 'count': 33}
{'_id': 'toilets', 'count': 32}
{'_id': 'community_centre', 'count': 29}
{'_id': 'waste_disposal', 'count': 26}
```

Number of Offices

In [81]:
```
off=len(db.TrentPrincemap.distinct('office'))
pprint.pprint("Number of Unique offices")
pprint.pprint(off)
```

```
'Number of Unique offices'
20
```

List out Offices

```
In [20]: topoffices  = db.TrentPrincemap.aggregate([
             {"$group" :
                 {"_id" : "$office",
                  "count" : {"$sum" : 1}}},
                   { '$sort' : { 'count' : -1 } }, { '$limit' : 20 }
                                  ])
         for k in topoffices:
             print(k)
```

```
{'_id': None, 'count': 766680}
{'_id': 'government', 'count': 45}
{'_id': 'yes', 'count': 41}
{'_id': 'association', 'count': 33}
{'_id': 'estate_agent', 'count': 26}
{'_id': 'lawyer', 'count': 26}
{'_id': 'company', 'count': 26}
{'_id': 'financial', 'count': 23}
{'_id': 'insurance', 'count': 15}
{'_id': 'tax_advisor', 'count': 14}
{'_id': 'employment_agency', 'count': 10}
{'_id': 'ngo', 'count': 9}
{'_id': 'accountant', 'count': 3}
{'_id': 'research', 'count': 3}
{'_id': 'notary', 'count': 2}
{'_id': 'bail_bond_agent', 'count': 2}
{'_id': 'architect', 'count': 1}
{'_id': 'newspaper', 'count': 1}
{'_id': 'energy_supplier', 'count': 1}
{'_id': 'educational_institution', 'count': 1}
```

Check how many Insurance offices are in the area and print the names

In [134]:
```
njm = db.TrentPrincemap.find({ "office": "insurance"},{"name":1});
num=0
for p in njm:
    pprint.pprint(p)
    num += 1
```

```
{'_id': ObjectId('61244af881d601a4ad6b9df5'),
 'name': 'MacLean Insurance Agency'}
{'_id': ObjectId('61244afa81d601a4ad6c2ce8'), 'name': 'Allstate'}
{'_id': ObjectId('61244b0281d601a4ad6e3a41'),
 'name': 'Insurance Solutions Concept Inc'}
{'_id': ObjectId('61244b0681d601a4ad6f2fd8'),
 'name': 'Peter Massaquoi Insurance Agency'}
{'_id': ObjectId('61244b0681d601a4ad6f3249'), 'name': 'State Farm'}
{'_id': ObjectId('61244b0681d601a4ad6f326d'), 'name': 'Karl Weidel'}
{'_id': ObjectId('61244b0681d601a4ad6f33f4'), 'name': "Avery's"}
{'_id': ObjectId('61244b0781d601a4ad6f4a06'),
 'name': 'Inter States Insurance Agency'}
{'_id': ObjectId('61244b0e81d601a4ad713bd4'), 'name': 'AAA Insurance'}
{'_id': ObjectId('61244b0f81d601a4ad714e3e'),
 'name': 'Munich Reinsurance America Inc.'}
{'_id': ObjectId('61244b0f81d601a4ad714e3f'),
 'name': 'Munich Reinsurance America, Inc.'}
{'_id': ObjectId('61244b0f81d601a4ad714e40'),
 'name': 'Munich Reinsurance America, Inc.'}
{'_id': ObjectId('61244b0f81d601a4ad714e45'),
 'name': 'Munich Reinsurance America, Inc.'}
{'_id': ObjectId('61244b1181d601a4ad719d3c'),
 'name': 'Ray Sayre Jr - State Farm Insurance Agent'}
{'_id': ObjectId('61244b1481d601a4ad723505'),
 'name': 'Harrah & Associates, Inc.'}
```

As per the analysis from above output it is clearly evident that NJM Insurance company is not listed. I found that Object Id for NJM Insurance doesn't have an element office. We need to add an element 'office':'insurance' to ObjectId('61205afadf613f107a142f8e').

Insert the element for NJM Insurance

In [135]:
```
ins=db.TrentPrincemap
insResult = coll.update_one({ "name": "NJM Insurance Group"}, {'$set': {'offic
e': 'insurance'}})
```

Check if NJM Insurance shows up in the search

In [87]:
```
## check if NJM Insurance company is listed
njm2 = db.TrentPrincemap.find({ "office": "insurance"},{"name":1});
num=0
for p2 in njm2:
    pprint.pprint(p2)
    num += 1
```

```
{'_id': ObjectId('6124292bafce4ba0f8d67142'),
 'name': 'MacLean Insurance Agency'}
{'_id': ObjectId('6124292eafce4ba0f8d70035'), 'name': 'Allstate'}
{'_id': ObjectId('61242937afce4ba0f8d90d8a'),
 'name': 'Insurance Solutions Concept Inc'}
{'_id': ObjectId('6124293aafce4ba0f8da0324'),
 'name': 'Peter Massaquoi Insurance Agency'}
{'_id': ObjectId('6124293aafce4ba0f8da0593'), 'name': 'State Farm'}
{'_id': ObjectId('6124293aafce4ba0f8da05a4'), 'name': 'Karl Weidel'}
{'_id': ObjectId('6124293aafce4ba0f8da0742'), 'name': "Avery's"}
{'_id': ObjectId('6124293aafce4ba0f8da1d51'),
 'name': 'Inter States Insurance Agency'}
{'_id': ObjectId('61242941afce4ba0f8dc0f27'), 'name': 'AAA Insurance'}
{'_id': ObjectId('61242941afce4ba0f8dc218c'),
 'name': 'Munich Reinsurance America, Inc.'}
{'_id': ObjectId('61242941afce4ba0f8dc218d'),
 'name': 'Munich Reinsurance America, Inc.'}
{'_id': ObjectId('61242941afce4ba0f8dc218e'),
 'name': 'Munich Reinsurance America Inc.'}
{'_id': ObjectId('61242941afce4ba0f8dc218f'),
 'name': 'Munich Reinsurance America, Inc.'}
{'_id': ObjectId('61242942afce4ba0f8dc33cb'), 'name': 'NJM Insurance Group'}
{'_id': ObjectId('61242943afce4ba0f8dc7087'),
 'name': 'Ray Sayre Jr - State Farm Insurance Agent'}
{'_id': ObjectId('61242946afce4ba0f8dd0851'),
 'name': 'Harrah & Associates, Inc.'}
```

Select max 15 healthcare departments in the area based on grouping by each healthcare type.

In [136]:
```
#Select max 15 healthcare departments in the area based on grouping by each he
althcare type.

#******************Top 15 Healthcare centers***********************
tophealthcares  = db.TrentPrincemap.aggregate([
    {"$group" :
        {"_id" : "$healthcare",
         "total" : {"$sum" : 1}}},
          { '$sort' : { 'count' : 1 } }, { '$limit' : 15 }
                              ])
for k in tophealthcares:
    print(k)
```

```
{'_id': 'rehabilitation', 'total': 4}
{'_id': 'dentist', 'total': 11}
{'_id': 'counselling', 'total': 1}
{'_id': 'podiatrist', 'total': 1}
{'_id': 'hospital', 'total': 7}
{'_id': None, 'total': 766890}
{'_id': 'alternative', 'total': 2}
{'_id': 'physiotherapist', 'total': 1}
{'_id': 'yes', 'total': 3}
{'_id': 'clinic', 'total': 7}
{'_id': 'doctor', 'total': 6}
{'_id': 'pharmacy', 'total': 27}
{'_id': 'laboratory', 'total': 3}
```

In [137]:
```
#---From the top 15 health care output derived from above code, it was noticed
that there is some bad data like 'yes' as a value of healthcare instead of typ
e.
baddata = db.TrentPrincemap.aggregate([{ '$match'  : { 'healthcare': 'yes' } }
,
                                { '$group' : { '_id' : '$healthcare', 'count':
{ '$sum' : 1 } } }
                                   ] )
for d in baddata:
     print("How many health cares have YES as a value instead of type?")
     print(d)
```

```
How many health cares have YES as a value instead of type?
{'_id': 'yes', 'count': 3}
```

Fix the bad data for healthcare

In [139]:
```
# Fixing the bad data healthcare issue
print("Lets fix the bad data issue")

# Update the healthcare in the document:use function update_many
coll=db.TrentPrincemap
updateResult = coll.update_many({'healthcare': 'yes'}, {'$set': {'healthcare':
'hospital'}})
```

```
Lets fix the bad data issue
```

In [140]:
```
#------Now Let's check again top 15 healthcare centres to see if 'yes' still e
xists
           #OR
#------Is the count of hospital increased from earler results because we moved
health cares with 'yes' into hospital type.


U_healthCare =db.TrentPrincemap.aggregate([
    {"$group" :
        {"_id" : "$healthcare",
         "total" : {"$sum" : 1}}},
          { '$sort' : { 'count' : 1 } }, { '$limit' : 15 }
    ])
for k in U_healthCare:
    print(k)
```

```
{'_id': 'dentist', 'total': 11}
{'_id': 'rehabilitation', 'total': 4}
{'_id': 'doctor', 'total': 6}
{'_id': 'pharmacy', 'total': 27}
{'_id': 'clinic', 'total': 7}
{'_id': 'laboratory', 'total': 3}
{'_id': 'counselling', 'total': 1}
{'_id': 'physiotherapist', 'total': 1}
{'_id': None, 'total': 766890}
{'_id': 'alternative', 'total': 2}
{'_id': 'hospital', 'total': 10}
{'_id': 'podiatrist', 'total': 1}
```

In [141]:
```
#----Check if 'healthcare': 'yes' is removed from the collection. This should
 give NONE.
x=db.TrentPrincemap.count_documents({ 'healthcare': 'yes'})
print(x)
```

```
0
```

Data Wrangling process is completed. All the corrected values are stored in JSON file.